DEVELOPING A LOCATION BASED REMINDER APPLICATION ON ANDROID PLATFORM

A thesis submitted in partial fulfillment of the requirements

for the degree of

Bachelor Of Technology

In

Electronics and Communication Engineering

By

Ashok Kumar Naik
Roll No- 109EC0225

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela

DEVELOPING A LOCATION BASED REMINDER APPLICATION ON ANDROID PLATFORM

A thesis submitted in partial fulfillment of the requirements

for the degree of

Bachelor Of Technology

In

Electronics and Communication Engineering

By

Ashok Kumar Naik
Roll No- 109EC0225

Under The Guidance Of

Prof. Sarat Kumar Patra

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela

NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

CERTIFICATE

This is to certify that the Theses "**Developing A Location Based Reminder Application on Android Platform**" submitted by Ashok Kumar Naik (109ec0225) in partial fulfillment of requirement for the award of Degree of **Bachelor of Technology** in Electronics and Communication Engineering at National Institute Of Technology, Rourkela (Deemed University) is an authentic work carried by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any university /institute for award of any degree or diploma.

Date: 11th May 2013

Prof Sarat Kumar Patra

Department Of Electronics and Communication Engineering

National Institute of Technology, Rourkela

NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

DECLARATION

I hereby declare that the project work entitled "**Developing A Location Based Reminder Application On Android Platform** " is a record of my original work, done under the guidance of **Prof Sarat Kumar Patra** at National Institute Of Technology Rourkela. Throughout this documentation wherever contributions of others are involved, every endeavour was made to acknowledge this clearly with due reference to literature. This work is being submitted as the partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics and Communication Engineering at National Institute of Technology, Rourkela for the academic session 2009 – 2013.

Date: 11th May 2013
NIT Rourkela

ASHOK KUMAR NAIK
109ec0225

ACKNOWLEDGEMENT

I would like to articulate our profound gratitude and indebtedness to those persons who helped me in completion of the project. First and foremost, I would like to convey our gratefulness to my project guide **Prof Sarat Kumar Patra** for his constant motivation and valuable suggestions throughout the project duration. I truly appreciate **Prof K K Mahapatra, Prof P K Tiwari**, **Prof S K Behera and Prof Ayaskant Swain** and all our **faculty members** for providing a solid background for my studies and research thereafter, which helped a lot to properly shape the problem and provided insights towards the solution.

I also want to express my sincere gratitude to **Prof S Meher, Head of the Department, Electronics and Communication Engineering** for allowing access to valuable facilities in the department.

I am also very thankful to my batch mates, who always encouraged and helped me in the successful completion of my thesis work. Also, I want to thank our lab-assistants and departmental office staffs for lending help whenever required. Last but not the least, I thank to all those friends who helped me in the course of this entire thesis work.

Date: 11th May 2013
NIT Rourkela

ASHOK KUMAR NAIK
109EC0225

CONTENTS

ABSTRACT

Most of the reminder applications available today in mobile phones are time and date based. In which the user has to save the time and date of when he wants to be reminded about in the reminder. If the reminder is ON, the device continuously tries to match the device time and date with the saved time and date, and the user will be alerted if it is a match. But in many cases the user will not be aware about the time and date, but he will be aware about the place where he wants the reminder. In this project, I have tried to design an application which gives alert about the reminder when he enters into the geographical region specified in the reminder. I have discussed about the android technology used in this application such as GPS and SQLite Database technology. I have also tried to provide additional features such as tone and vibration to catch user attention so that he does not miss the alert.

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction to Android

Open Source Platform

Android

Setting up the Eclipse IDE and Android SDK

1.1 Open source platform

Open source is a program in which the source code is available to the general public via free license for use and modification from its original form. Open source code is typically created as a collaborative effort in which programmers improve the code and share the changes within the community.

The rights attached to the program must apply to all whom the program is redistributed without requiring any additional license. The license must not place restrictions on other software that is distributed along with the licensed software i.e. the license should not impose a condition that all other programs distributed on the same medium must be open-source software.

1.2 Android

Android is an open source platform licensed under business-friendly license called Apache, which provides “free software licenses” to software products. This enable users to use , modify, redistribute Android. The entire stack, from low-level Linux modules to native libraries, and from the application framework to complete applications, is totally open. Also some open source third party libraries such as SQLite ,WebKit,OpenGL has been added to android.

1.2.1 History

Android was the brainchild of Andy Rubin, one of the cofounders of “Android Inc” (others are Rich Miner, Nick Sears, Chris White). He decided to produce a better smartphone operating system which can compete with the other mobile operating systems of that time like symbian and windows. But he faced some financial problem that time and in 2005, Google buys “Android Inc” owning it wholly as a subsidiary of it. The key employee of Android Inc started working under Google and developed this platform purely based on Linux kernel.

In 2007, the Open Handset Alliance (OHA), a consortium of more than 50 technology companies including Google, HTC ,Samsung , QualComm, TI was announced and Android was officially announced open sourced.

1.2.2 Evolution

In 2008, the first ever software development kit of Android was released as “Android SDK 1.0”. The first commercially available phone to run Android was the HTC Dream, released on October 22, 2008. Since 2008, Android has no doubt gone through numerous updates which have incrementally improved the operating system in terms of new features, processing power,

eye-catching UI and bug fixing abilities from the previous releases. Each new release is being named in alphabetical order after a dessert or sugary treat. The following table lists all the version of Android releases till today

Platform Version	**API Levels**	**Version Code**
Android 1.0 Android 1.1	1 2	Base
Android 1.5	3	Cupcake
Android 1.6	4	Donut
Android 2.0 Android 2.0.1 Android 2.1	5 6 7	Éclair
Android 2.2	8	FROYO(Frozen Yogurt)
Android 2.3 to Android 2.3.4	9 ,10	Ginger Bread
Android 3.0 Android 3.1 Android 3.2	11 12 13	Honey Comb
Android 4.0 to Android 4.0.4	14,15	Ice Cream Sandwich
Android 4.1 Android 4.2	16,17	Jelly Bean

Table 1.1 Lists of Android versions till date

1.2.3 Features

- Android is a hardware reference design which clearly describes the capabilities needed for a mobile device to support the software stack.
- It has a "Linux operating system kernel" which provides low-level interface with the hardware, memory management, and process control, optimized for mobile devices.
- Open-source third party libraries for application development such as SQLite, WebKit, OpenGL, and a media manager.

- A run time used to execute and host Android applications, including the Dalvik virtual machine and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on mobile devices.
- An application framework that agnostically exposes system services to the application layer, including the window manager and location manager, content providers, telephony, and sensors.
- A user interface framework used to host and launch applications.
- Preinstalled sample applications available.
- A software development kit used to create applications, including tools, plug-ins, and documentation.

1.3 Setting up the Eclipse IDE and Android SDK

The application has been developed using Eclipse tool along with android SDK platform.

❖ **Installation Procedure**

First we need to install any version of Eclipse IDE (Eclipse 4.2 JUNO has been used here) which provides the editor window for writing java codes. Even though it is not compulsory to use Eclipse's editor it is a better option to use it.

Now go to the "install new software**"** option inside help menu. Put the web address https**://dl-ssl.google.com/android/eclipse/** in the work with box. It will show the Android plug-in in the list. Check this option and install it. This plug-in is used for connecting eclipse and Android SDK.

Next download and unzip any version of Android SDK. It is a better option to use Android 2.2 (Froyo) for beginners. Now go to Window>Preference and then Click "Browse**"** and

navigate to the folder into which you unzipped the Android SDK then click **Apply.** The list will then update to display each of the available SDK targets, Click **OK** to complete the SDK installation.

Now we are ready to create our first Android application i.e. if we select new > project and there should be an option **Android Project**.

Chapter 2
Application Development on Android Platform

The software Stack

Factors Should Be Considered

2.1 The Software Stack

Before starting to make application for Android, we should have some idea about the android operating system framework. Understanding the layout of the system will help shape your understanding about what we can or cannot do easily with Android. The android operating system is like a cake, consisting of different layers. These layers are not clearly separated from each other, rather they are overlapped onto each other. Each layer has its own purpose and property. The layers are Linux kernel, libraries, the Android run-time, application framework and the application layer. The following figure clearly explains the various layers available in Android software stack.

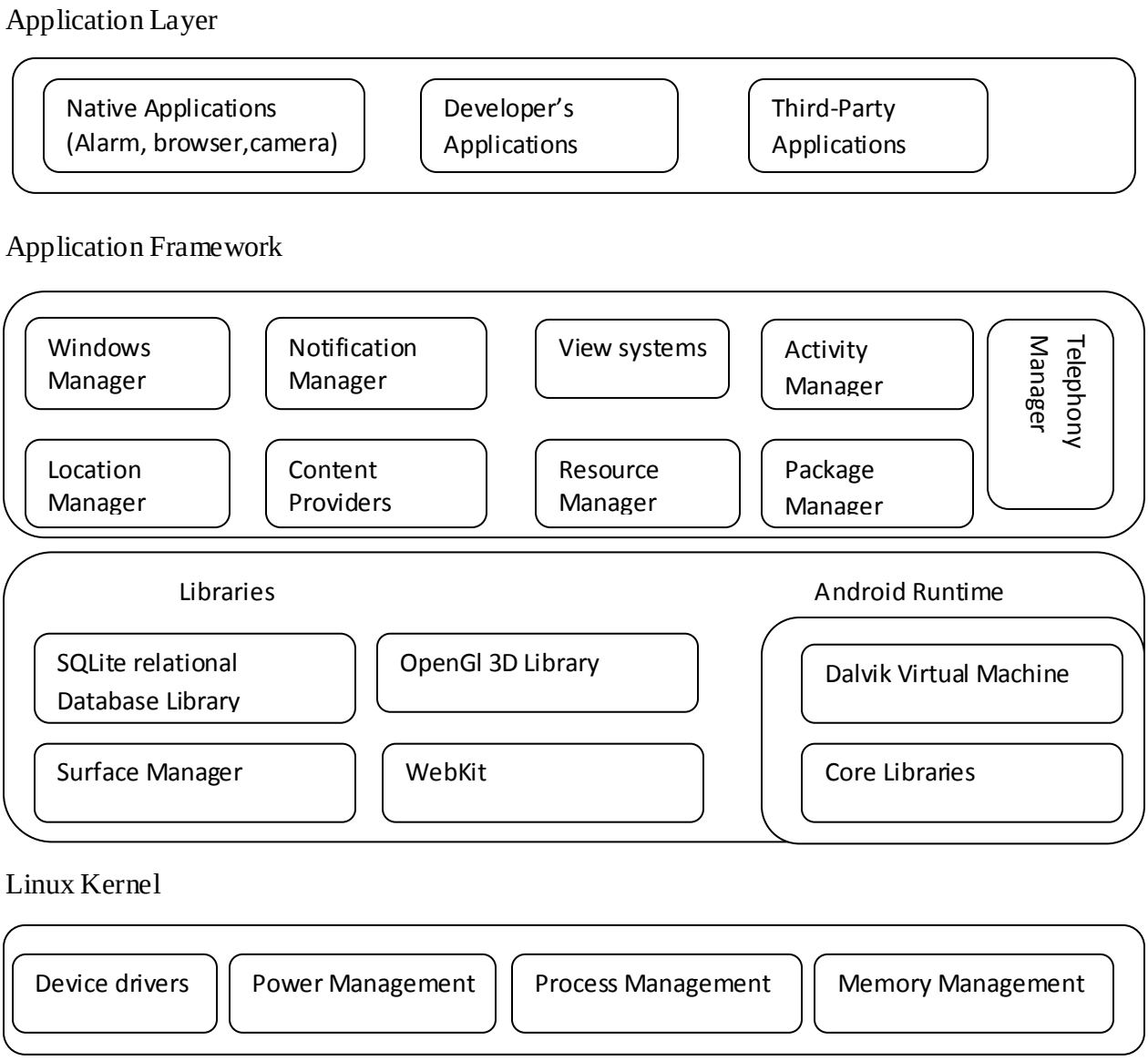


FIGURE 2.1 Components of Android Operating System

LINUX KERNEL

This is the core of all services. Its services include device drivers, memory management, process management and power management. it also provides an abstraction layer between device and remainder of the stack. It uses android version 2.6 for the services.

LIBRARY

This runs on the top of kernel and contains some C/C++ libraries system C library,3d graphics library, media library ,relational database library SQLite, the underlying 2D library.

ANDROID RUNTIME

The Android run time is the engine which includes the core libraries and the Dalvik virtual machine. It powers the applications and, along with the libraries which forms the basis for the application framework.

- **Core libraries**

  While Android development is done in Java, Dalvik is not a Java VM Application framework.. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android-specific libraries.

- **Dalvik virtual machine**

  Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

APPLICATION FRAMEWORK

The application framework provides the classes used to create Android applications and offers developers the ability to take advantage of the device hardware, notification, background services, set alarms etc. It also manages users to set and manage the User Interfaces and resources.

APPLICATION LAYER

All the applications, native, developers' and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android run time, using the classes and services made available from the application framework.

2.2 Factors Should Be Considered

To be able to design an efficient Android application in terms of speed, power and resource usage, we should take care of the following factors.

2.2.1 Hardware-Imposed Design Consideration

Mobile devices have relatively:

- Low processing power
- Limited RAM
- Limited permanent storage capacity
- Small screens with low resolution
- High costs , slow data rates ,latency associated with data transfer
- Limited battery life

Each new generation of phones improves many of these restrictions. In particular, newer phones have dramatically improved screen resolutions and significantly cheaper data tariffs. However, given the range of devices available, it is good practice to design to accommodate the worst-case scenario.

2.2.2 Efficiency

Particularly for mobile devices, generally emphasis is given on small device size and long battery life over potential improvements in processor speed. Over years, increase in transistor density on the devices (Moore's law, doubles every 2 year) indicates smaller device size and more power efficient without significant improvement in processor power. We should focus on optimizing the code so that it runs quickly and responsively, assuming that hardware improvements over the lifetime are unlikely to do any favors.

2.2.3 Limited Capacity

It is obvious that most of the available storage on a mobile device is likely to be used to store music and movies, most devices offer relatively limited storage space for your applications. Android devices offer an additional restriction in that applications must be installed on the internal memory. As a result, the compiled size of your application is a consideration .Careful consideration should be taken on how to store application data.

2.2.4 Low Speed and High Latency

The mobile Web unfortunately isn't as fast, reliable, or readily available as we'd often like, so when you're developing your Internet-based applications it's best to assume that the network connection will be slow, intermittent, and expensive. Designing for the worst case ensures that you always deliver a high-standard user experience.

2.2.5 Cost

Services like SMS, some location-based services, and data transfer can sometimes incur an additional tariff from your service provider. It's obvious why it's important that any costs associated with functionality in your applications be minimized.

Cost incurred in an application can be minimized by

- Transferring as little data as possible
- Caching data and GPS results to eliminate redundant or repetitive lookups
- Stopping all data transfers and GPS updates when your activity is not visible in the foreground and if they're only being used to update the UI.
- Keeping the refresh/update rates for data transfers (and location lookups) as low as possible.

Chapter 3
Android Building Blocks Used

Activity and Intents

Databases

Adapters

Services and AsyncTasks

Location Based Services (LBS)

AlertDialog and ProgressDialog

Notifications

3.1 Activity and Intents

An “Activity” in android is basically a screen, which the user sees on the device at one time, such as dial the phone, take a photo, click a button etc. The screen might be filled completely by User Interface element or might be half-filled or floating on top of other screen. As such, activities are the most visible part of your application. An application typically has multiple activities which are loosely bound to each other and the user flips back and forth among them and this may also involve passing of data between them. Whenever the user goes to another activity, the previous activity is stopped, but the system preserves the activity in a stack called “back stack”. Each Activity has a specific view and performs different tasks according to the

state of the application. The task performed during a state can be better understood by looking at the activity life cycle.

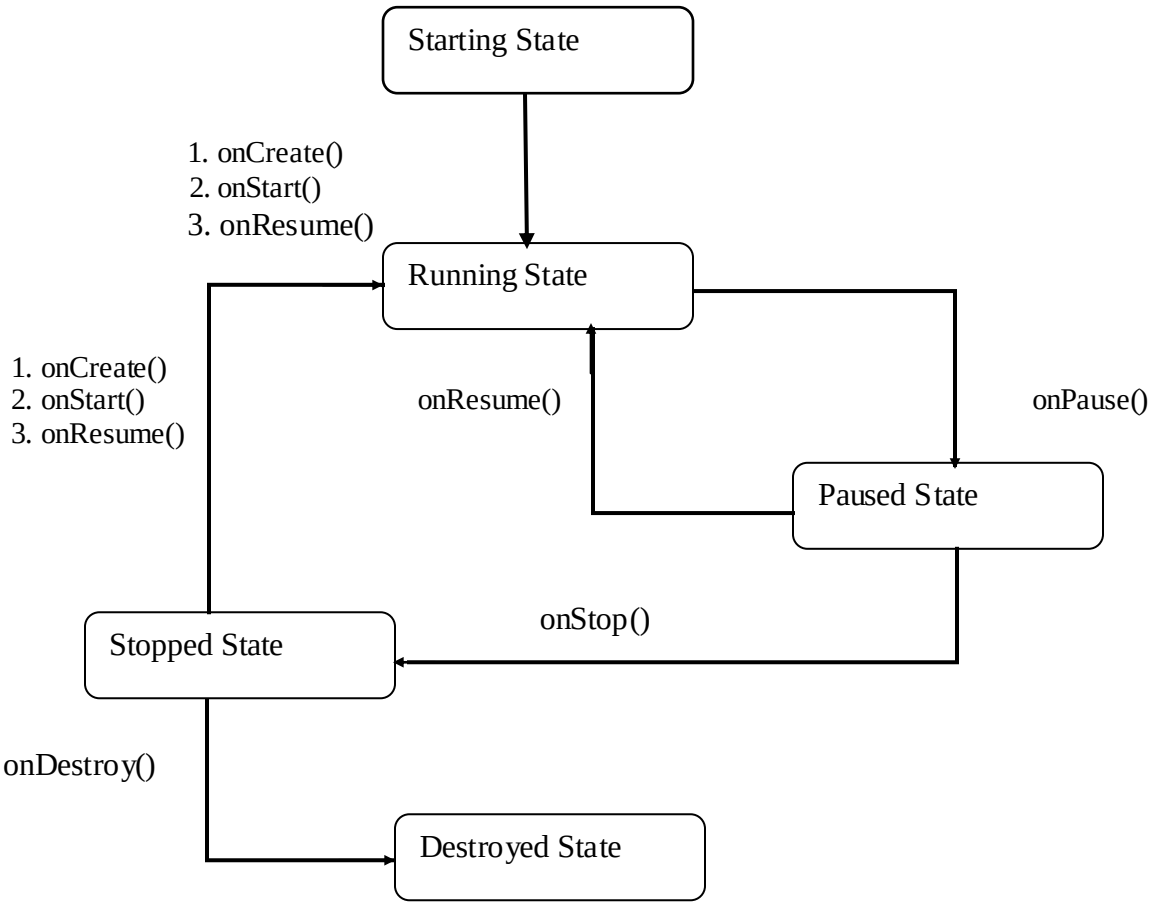


FIGURE 3.1 Activity Life Cycle

Whenever an Activity is launched, it first comes to the starting state. Then the methods onCreate(),onStart(),onResume() are run sequentially and the activity comes to the running state. During this time, if another Activity comes to the foreground, but the previous Activity is visible, the onPause() method is run and the Activity goes to the Paused state. If the Activity is no longer visible the onstop() method is called and the activity goes to Stopped state. But, it is important is remember that an Activity never comes to the Stopped state directly; it always comes through the Paused State. If the Activity is finished or destroyed by the system for the purpose of saving

memory then the onDestroy() method is called and the Activity is shut down. Similarly, to go back to Running state from Paused state, it calls the onResume() method. But for Stopped state and Destroyed state they have to create the Activity again.

INTENTS are messages that are sent among the major building blocks. They trigger an activity to start up, tell a service to start or stop, or are used simply to broadcast. Intents are asynchronous; meaning the code that sends them doesn't have to wait for them to be completed. Intent could be explicit or implicit. In an explicit intent, the sender clearly spells out which specific component should be on the receiving end. In an implicit intent, the sender specifies the type of receiver. This involves the use of receiver and an "Intent Filter" which is used as a action name. Thorough Intents, we can also pass data between the Activities.

3.2 Databases

Android provides 3 ways of data storage facility

- saving key-value pair in simple SharedPreferences file
- saving arbitrary files in Android's file system
- Saving in databases managed by SQLite

Database is useful for saving structured or repeating data such as contact information, where each information has similar set of attributes. The SQLite database library, a well regarded relational database management system (RDBMS). It has been implemented as a compact C library that's included as part of the Android software stack. By being implemented as a library, rather than running as a separate ongoing process, each SQLite database is an integrated part of

the application that created it. This reduces external dependencies, minimizes latency, and simplifies transaction locking and synchronization.

Android databases are stored in the /data/data/<package name>/databases folder on your device (or emulator). By default all databases are private, accessible only by the application that created them. Every application can create its own databases over which it has complete control. Android provides an elegant interface for the application to provide a "connection" to the database, creating the connection if it doesn't already exist. This is provided by a class called "SQLiteOpen Helper" from the android framework.

The SQLiteOpenHelper class offers a high-level interface that's much simpler than SQL. As we know that that most applications use databases for only four major operations, SQLiteOpenHelper offers the following pre-defined methods

insert() : To Insert one or more rows into the database

query() : To request rows matching the criteria specified.

update() : To replace one or more rows that match the criteria specified

delete() : To delete rows matching the criteria specified.

For all other valid SQL commands, call the execSQL method to run the SQL code by writing the command in String format.

Why SQLite, instead of SQL directly?

From a security point of view, an SQL statement is a prime candidate for a security attack on your application and data, known as an "SQL injection attack". That is because the SQL statement takes user input, and unless you check and isolate it very carefully, this input could embed other SQL statements with undesirable effects. Data is read from the Database through the Cursor using the query() method and inserted through Content Values.

3.3 Adapters

ADAPTERS are the bridge between data and view (especially a list View). They often carry the underlying data provided mainly by an Array or a database and display it in the child views of the parent view (List View in most cases). There are several kinds of adapters provided by Android such as simpleCursorAdapter, spinnerAdapter, listAdapter, arrayAdapter and many more.

The Array Adapter uses generics to bind an Adapter View to an array of objects of the specified class. By default the Array Adapter uses the String value of each object in the array to create and populate Text Views. Alternative constructors enable you to use more complex layouts, or you can extend the class to use alternatives to Text Views as shown in the next section.

The Simple Cursor Adapter attaches Views specified within a layout to the columns of Cursors returned from the query of a database. We have to specify a XML layout for each row and map the data of each column of the cursor to the different views within the layout. The adapter creates a new view for each of the entry in the cursor.

The simpleCursorAdapter has been used for this project to populate each child View of the list View. The syntax of a simpleCursorAdapter looks like this

```
SimpleCursorAdapter adapter;
adapter = new SimpleCursorAdapter( Context, row layout , To, From)
```

The parameters are Context which is the context of current class, row layout which is the layout for each child view in the list. “To” specifies the integer ids in the row layout and “From” specifies the column names.

3.4 Services and AsyncTasks

Unlike Activities, which present a rich graphical interface to users, Services run in the background. They are the perfect means of performing ongoing or regular processing and of handling events even when your application's Activities are invisible or inactive, or have been closed.

- Service does not have a UI and prohibits the access to UI (if tried it will give a force close error)
- Services are started, stopped, and controlled from other application components, including other Services, Activities, and Broadcast Receivers.
- Services always have higher priority than inactive or invisible Activities, making them less likely to be terminated by the run time's resource management.
- Services are launched on the main Application thread, meaning that any processing done in the onStartCommand handler will happen on the main GUI thread.

If you return this value, onStartCommand will be called any time your Service restarts after being terminated by the run time. Note that on a restart the Intent parameter passed in to onStartCommand will be null. This mode is typically used for Services that handle their own states, and that are explicitly started and stopped as required (via startService and stopService).

The AsyncTask class offers a simple, convenient mechanism for moving your time consuming operations onto a background thread avoiding FORCE CLOSE error. It offers the convenience of event handlers synchronized with the GUI thread to let you update Views and other UI elements to report progress or publish results when your task is complete.

AsyncTask handles all of the thread creation, management, and synchronization, enabling you to create an asynchronous task consisting of processing to be done in the background and a UI update to be performed when processing is complete.

The AsyncTask skeleton looks like this

```
private class MyAsyncTask extends AsyncTask<Location, Integer, String> {

    @Override
    protected void onProgressUpdate(Integer... progress) {  }

    @Override
    protected String doInBackground(Location... parameter) {  }

    @Override
    protected void onPostExecute(String... result) {  }

    }
```

The doInBackground method runs the time consuming task in separate thread. While the process is going on we can update about the progress through publishProgress method from within the doInBackground method. Here we publish an integer value which can be used in the onProgrssUpdate method to inform the user about the progress. The doInBackground method also returns some value on completion (string value in this case) which is caught by the onPostExecute method to perform further operation on the output. The two handlers onPostExecute and onProgrssUpdate are synchronized with the main GUI thread so we can access all the User Interface element from here and can modify them. However the doInBackground method runs in the background thread and prohibits access to UI elements.

3.5 Location Based Services (LBS)

Android also provides Location based services which include Google MAP API through which we can display maps in our app, GPS and cell based location technology to get the co-ordinate of the current location, forward and reverse geocoding technology to convert back and forth between latitude and longitude values and real-world addresses.

Location based services are system services, to use it we first have to get an instance of it through LOCATIONMANAGER. Location Manager controls all kinds of location based service in the application.

The syntax for accessing the Location based services is

```
String service = Context.LOCATION_SERVICE;
LocationManager location;
location  = (LocationManager)getSystemService(service);
```

Before we can use the location based service's hardware, we should not forget to add the following two permissions to the Manifest file.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_INTERNET"/>
```

To make the application location sensitive to user movement, we have to create an instance of Location listener. The listener will update us with new location when we register it using the location Manager. From the location provided by the location listener we can get the latitude and longitude value using the proper method. After that, we can extract the address using the Geocoder object.

3.6 AlertDialog and ProgressDialog

There are basically two types of ProgressDialog used in android one spinner style and other is the horizontal style. These are to display to the user when there is a background processing is going on. Depending upon the progress, the horizontal style dialog fills its rectangular space. But spinner style keeps rotating until we haven't dismissed it. These are pretty easy to implement.

AlertDialog on the other hand are used to alert the user when some specific goal is achieved. AlertDialog normally appear as transparent window and they help user to confirm action and to display error and warning messages.

An AlertDialog can contain

1. Three buttons to perform different task
2. A layout which can contain a list, checkbox or a group of radio buttons
3. Text entry box and Title and message to inform the user

3.7 Notifications

Notifications is the preferred way for invisible application components (Broadcast Receivers, Services, and inactive Activities) to alert users that events have occurred that may require attention. They are also used to indicate ongoing background Services. Notifications Are the best ways to convey information because they additional feature they provide. Notification provide the following additional feature

1. Status bar icon with ticker text

2. Extended window as notification status drawer

3. Additional phone effects such as vibration, playing audio, flashing the device LEDs

Chapter 4
Results and Discussion

Device Description

Result with Discussion

4.1 Device Description

The application has been tested on the Dalvik Debug emulator which comes with the Android SDK .The device used was an Android 2.2 emulator with ARM processor of skin type WVGA 800 and keeping Storage Disk size 100 MB .

As this application needs original GPS sensor to locate the device position, only the virtual device is not sufficient to test this application. So a real Android device (an android OS based smart phone) “SAMSUNG GT-i9070” has been used to test this application. So the outputs are from both the Emulator and the smart phone.

4.2 Results with description

The following figures are the different screens of my Reminder application and the description of the action they perform are given along with them.

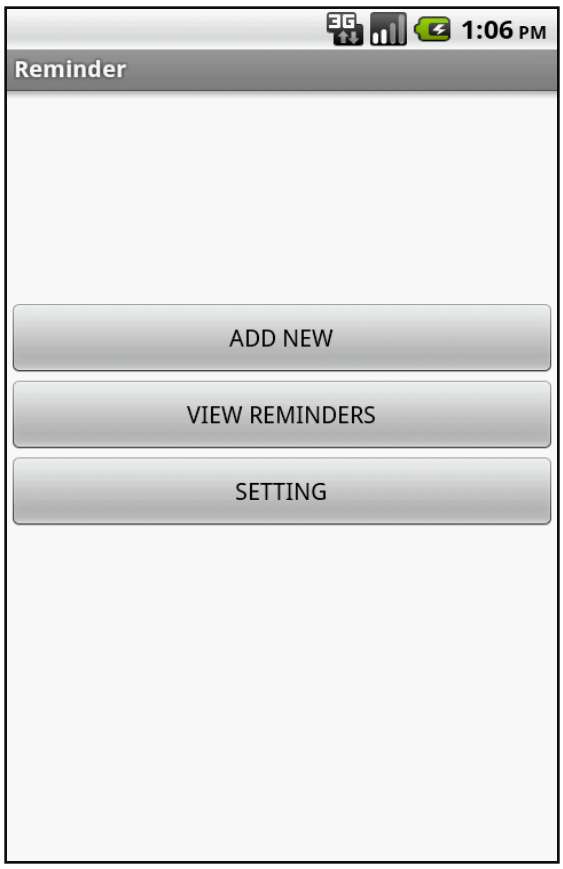


FIG 4.1 Home Screen

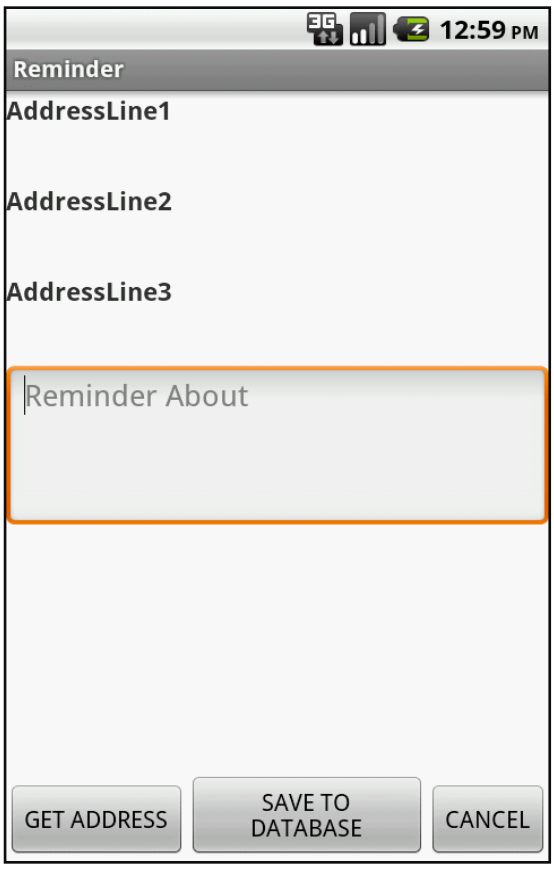
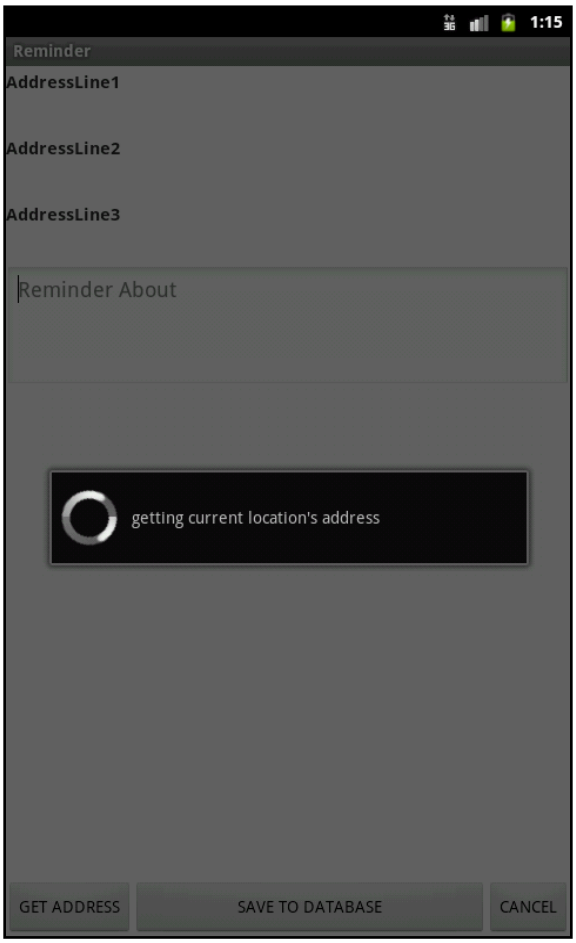


FIG 4.2 Add Reminder Screen

Figure 4.1 represents the home screen of my Reminder application. This is the first screen the user will see upon starting the application. It contains three buttons named Add Reminder, View Reminder and Setting. Each button leads to three different activities and performing actions as per the names written on them.

Figure 4.2 shows the Add Reminder screen which is started on clicking the corresponding button on the home screen. It contains three Text Views to hold 3 address line data and one Edit Text to let the user write the task he wants to be reminded about. It also holds three buttons “Get

Address" to get the current device's location's address, "Save to Database" to save the addresses and tasks to database and "cancel" to go back to previous activity.



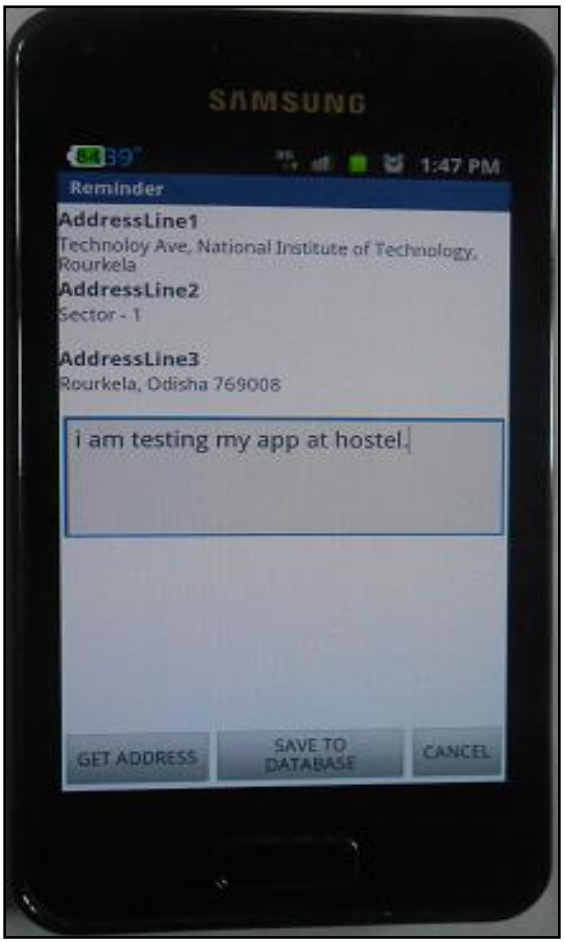


FIG 4.3 addressing searching on progress FIG 4.4 Address found

Figure 4.3 shows the screen when the searching process for address is going on. A transparent progress dialog of spinner style appears on the front screen. Figure 4.4 shows that the GPS has found the address. Now the user can write the task and click the "save to database" button to save all these results.

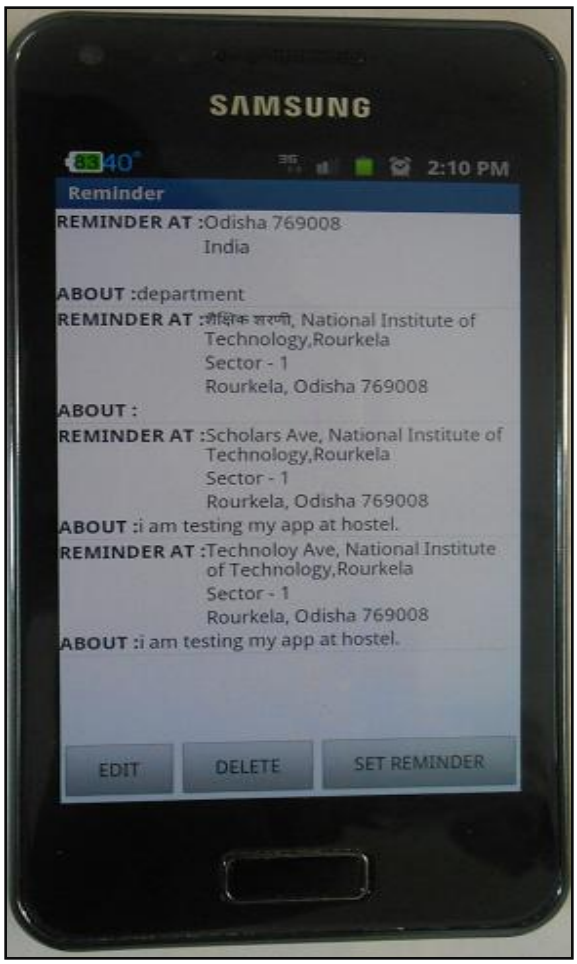


FIG 4.5 View Reminder screen

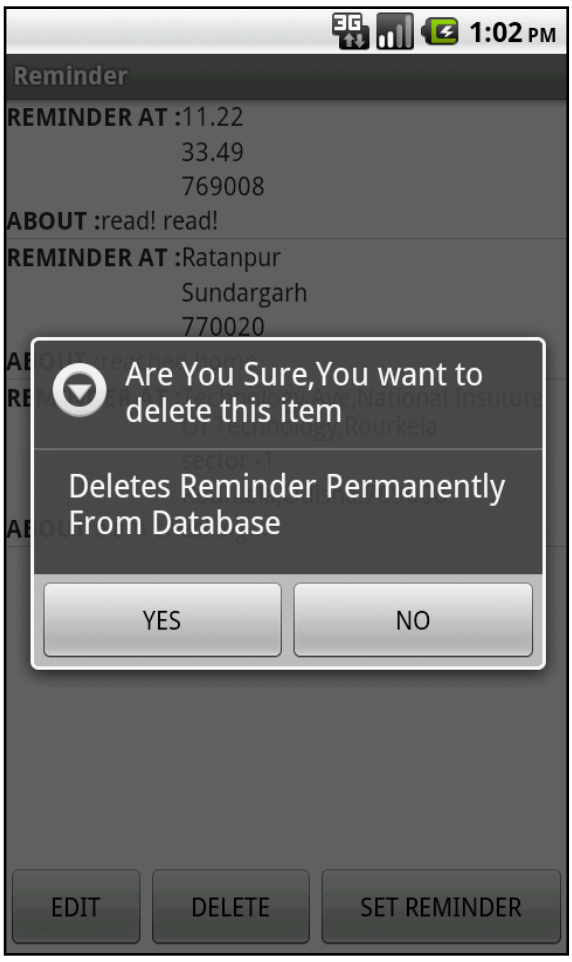


FIG 4.6 Delete Reminder

Figure 4.5 lists all the saved reminder in the database in a list format. Below the screen, it holds three buttons Edit, Delete and Set Reminder. To get the functionality of the buttons, first click on the reminder item upon which we want to perform action and then click the button.

Figure 4.6 shows the result of clicking the "Delete" button. It displays an AlertDialog with the title "are you sure, you want to delete this item" and message "Deletes reminder permanently from database". It also holds two buttons "YES" and "NO" to perform the corresponding tasks.

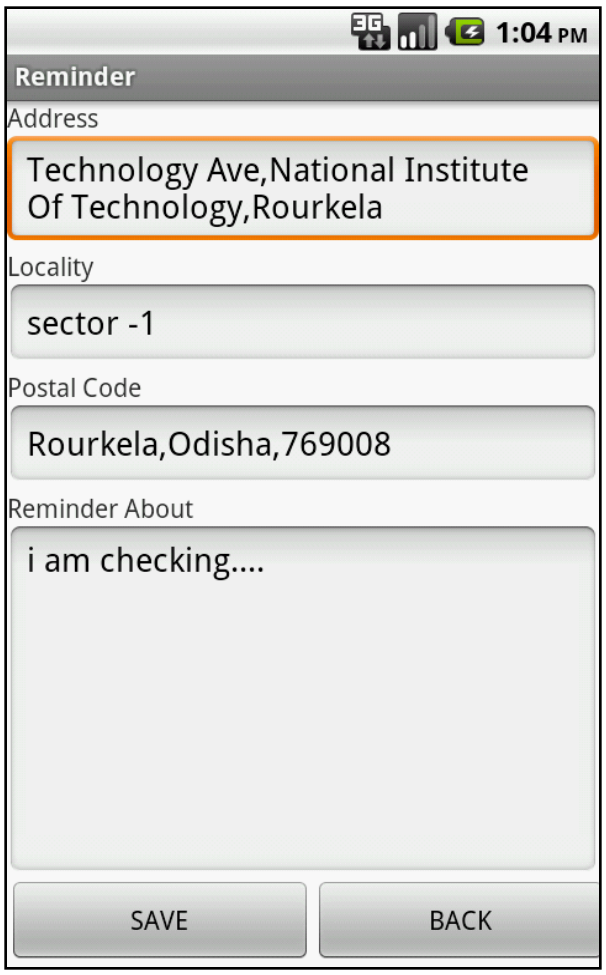


FIG 4.7 Edit screen

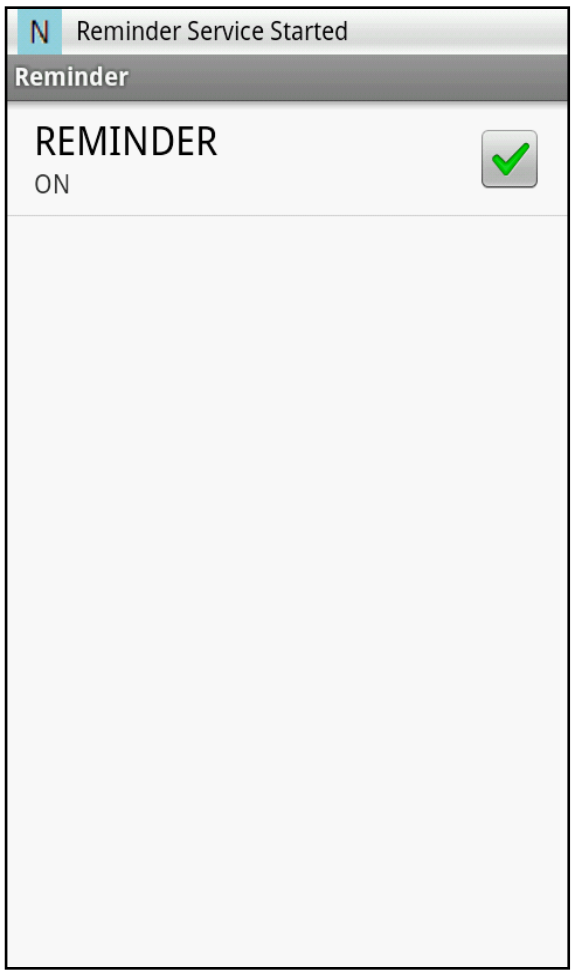


FIG 4.8 Set Reminder

Figure 4.7 displays the Edit screen which is started from the View screen. It displays all the data in their respective Edit Text views so that the user can modify the data. But the address entries should not be changed if it is not exactly an address given by GPS. Because those addresses are compared with the dynamic address found by GPS to alert Reminder. Upon clicking the SAVE button the edited reminder is updated on the previous row of list.

Figure 4.8 shows the Set Reminder screen which contains a check box. Upon checking this box the reminder is started and this is informed by the Notification with its ticker text "Reminder Service Started".

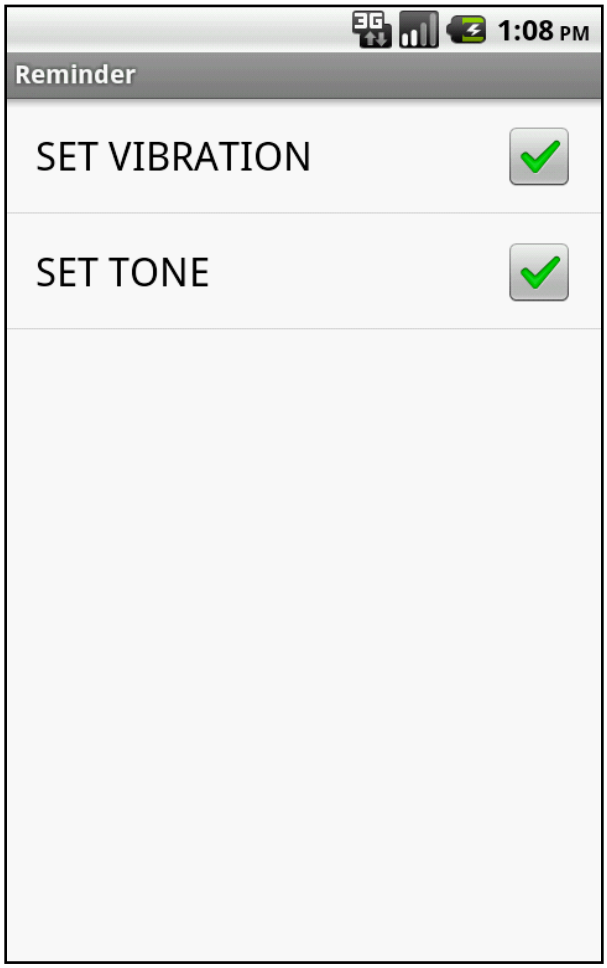


FIG 4.9 Setting screen

FIG 4.10 Expanded Notification window

Figure 4.9 gives the Set reminder window which has two check boxes named “Vibration” and “Tone”. Checking the boxes will play the tone or create vibration for four seconds (which has been set in the code).

Figure 4.10 gives the extended Notification window which appears on dragging down the notification bar. It holds additional information about the reminder. As in this case, it shows the row number of the active reminder. It also shows that the GPS is on.

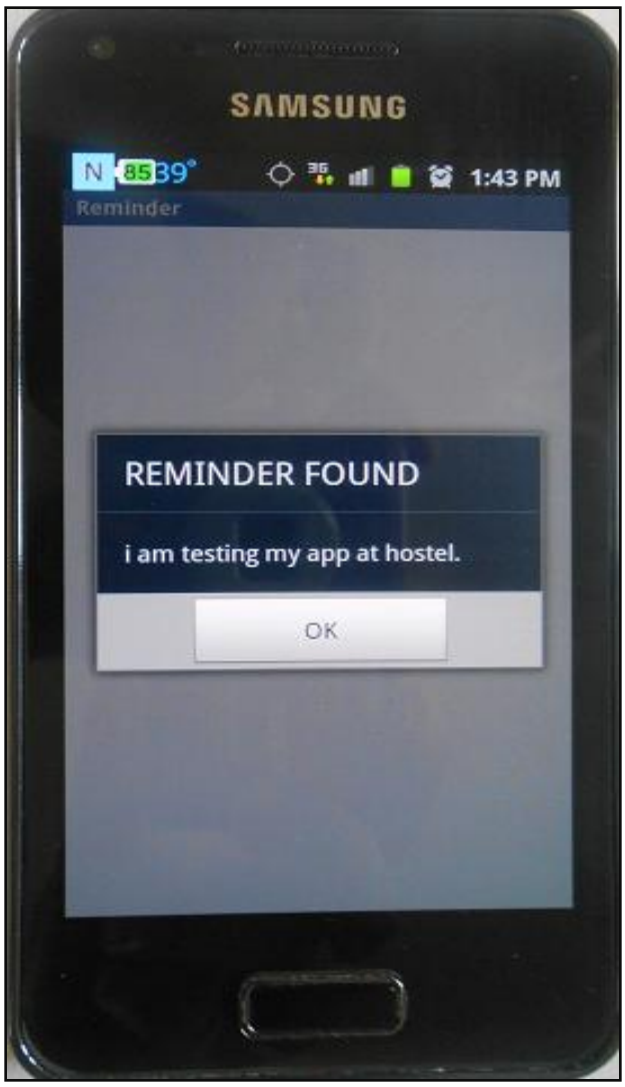


FIG 4.11 Final reminder alert

This is the final AlertDialog appears when there is a match among the saved address and the address found by the GPS.

It displays a title "REMINDER FOUND" with the message as saved in reminder.

The button "OK" performs several tasks on clicking. It cancels the dialog, stops the service cancels the Notifications and unchecks the "Set Reminder" button.

Chapter 5
Conclusion

CONCLUSION

- A Location Based Reminder application on Android platform was successfully developed which can save one reminder at a time.
- It has been tested by saving addresses at different locations and it gave proper output at proper place.
- The Vibration and Tone setting are also checked. Checking the tone check box plays the tone installed in the application for the specified duration and checking the vibration check box also vibrates the device as specified in the program.

- As Location based application consume more power, effort has been made to reduce power consumption by setting a time interval of 20 seconds and a distance interval of 50 meters has been set between two consecutive location updates.
- As it is GPS based application, it may not work inside a room or during cloudy weather. Its response is also little bit slower.

REFERENCES

[1] Wei Meng Lee , "Beginning Android Application Development"

[2] Reto Mier, " Professional Android Application Development"

[3] Marko Gargenta , "Learning Android" , Oreilly publication

[4] B. Travis, "New Boston Series – Android Application Development video Lectures"

[5] Web Resources : "http://developer.android.com" , "http://stackoverflow.com"

[6] Jason Wei, " Android database programming",